Subgoal-Guided Policy Heuristic Search with Learned Subgoals

Jake Tuero, Michael Buro, Levi Lelis

Department of Computing Science, University of Alberta

Introduction

Policy Tree Search:

- A class of search algorithms which uses a *policy* to guide the search
- A policy is a probability distribution over the set of actions
- These algorithms provide guarantees on the number of expansions required to solve a given problem, based on the quality of the policy

The Bootstrap Search-and-Learn Process:

- Randomly initialized neural models encoding the heuristic and the policy are used to iteratively solve a subset of the training problems
- If the search cannot solve problems within a search budget, the resulting trees are discarded
- If at least one problem is solved, the models are optimized on the solution trajectories found

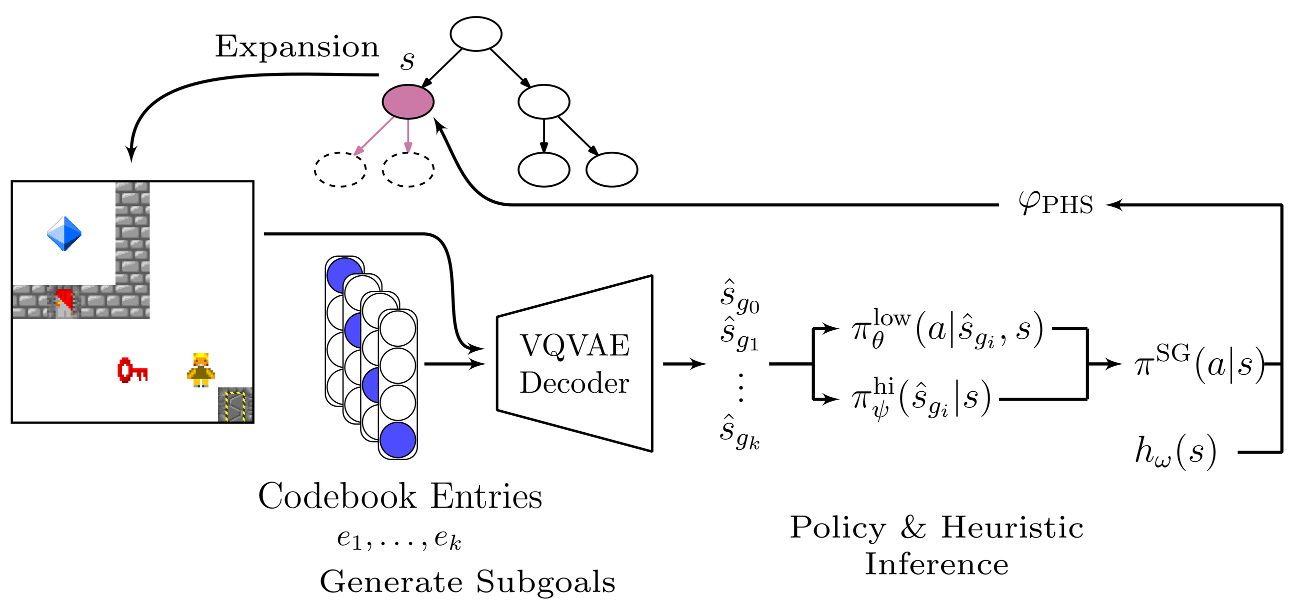
Motivation

- Search during the online bootstrap process generates a lot of data, but none of it is used if the search terminates prematurely
- Existing methods use a single policy which can become overburdened on complex domains; problems can usually be decomposed into subtasks/subgoals

Problem Statement

- This work focuses on solving single-agent deterministic search problems, using minimal domain knowledge
- Given a set of problem instances, the objective is to solve them while minimizing the *total search loss*

Subgoal-Guided Policy Heuristic Search



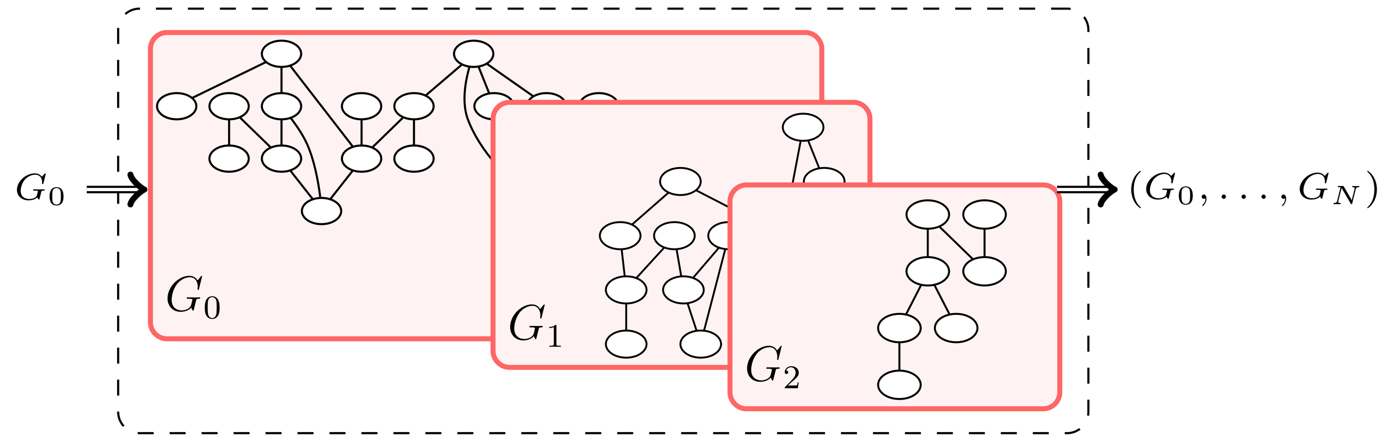
Contact

Jake Tuero
tuero@ualberta.ca

Paper **LinkedIn**

Training from Non-Solution Trees

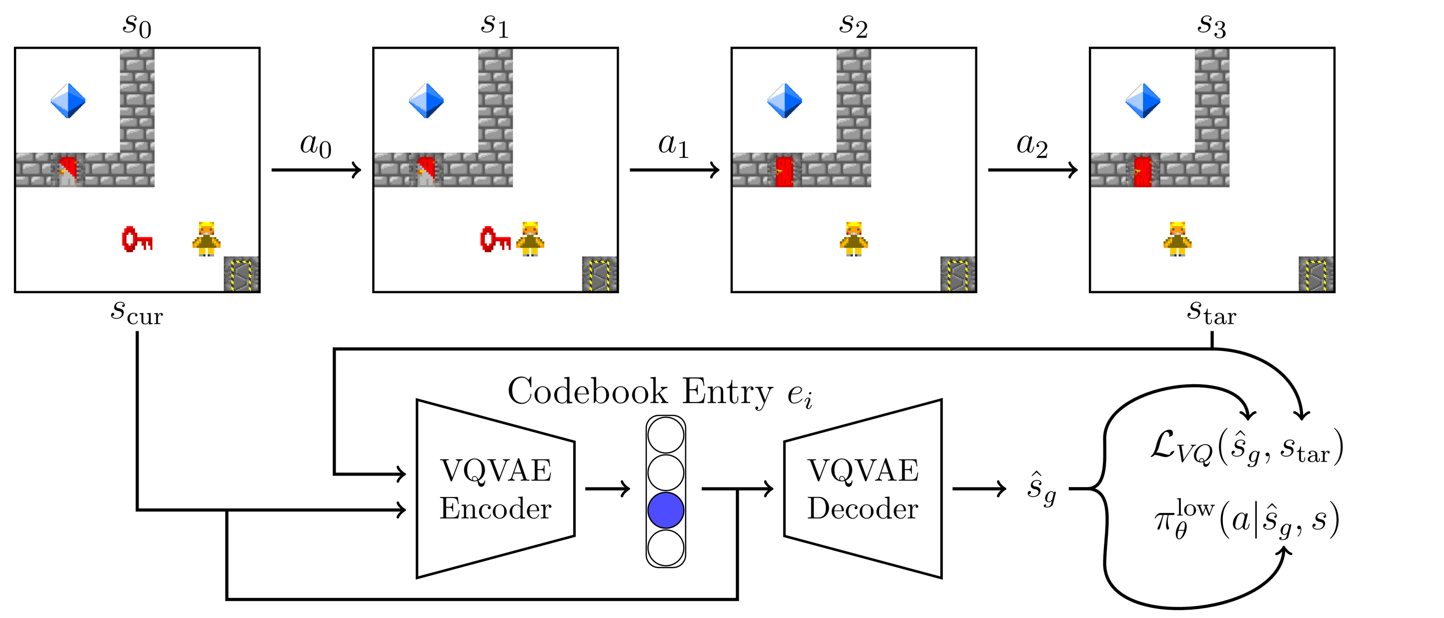
(i) Louvain Algorithm



(ii) Sample training data for VQVAE

1. Let $(G_0, G_1, \cdots, G_N)$ be the output of the Louvain algorithm.
2. Select graph $G_k = (V_k, A_k)$ for a given cluster level k in $[0, N]$ from $(G_0, G_1, \cdots, G_N)$.
3. Sample edge (v_k, v_k') in from A_k.
4. Sample s_{cur} and s_{tar} from v_k and v_k' respectively such that s_{cur} and s_{tar} are in V_0.

(iii) Reconstruct path in G_0 from s_{cur} to s_{tar},
Update subgoal generator and low-level policy



Training from Solution Trees

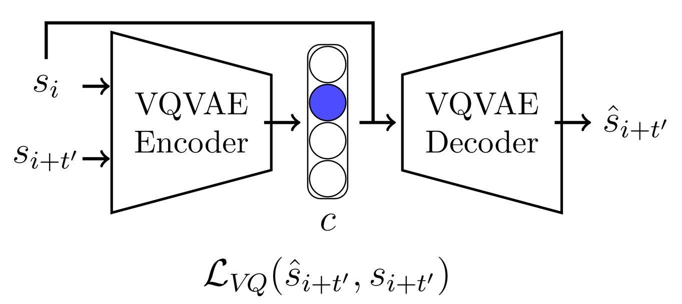
(i) Update heuristic model

1. Generate training set $\mathcal{D} = \{(s_i, t-i)\}_{i=0}^{t}$ given $s_0, a_0, s_1, \ldots, s_{t-1}, a_{t-1}, s_t$
2. Update h_ω with $\mathcal{D}$

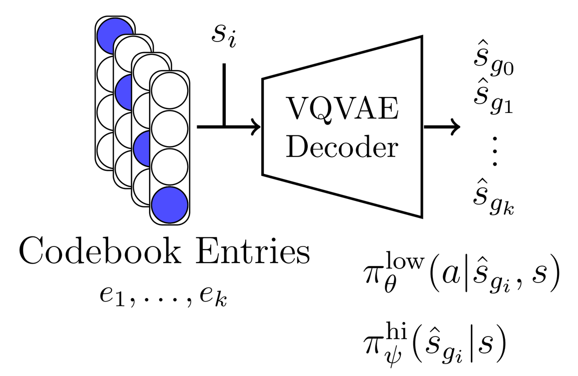
(ii) Segment solution trajectory

1. Calculate running mean and variance of sampled trajectories from (b.ii)
   μ = mean trajectory length from (b.ii)
   σ^2 = variance of the trajectory lengths from (b.ii)
2. Sample segment length
   $t' \sim \mathcal{N}(\mu, \sigma^2)$
3. Segment solution trajectory into subtrajectories of length t'
   $s_0, a_0, s_1, \ldots, s_{t'}$
   $s_{t'}, a_{t'}, s_{t'+1}, \ldots, s_{2t'}$
   $\vdots$
   $s_{(t-t')}, a_{(t-t')}, s_{(t-t'+1)}, \ldots, s_t$

(iii) Update subgoal generator

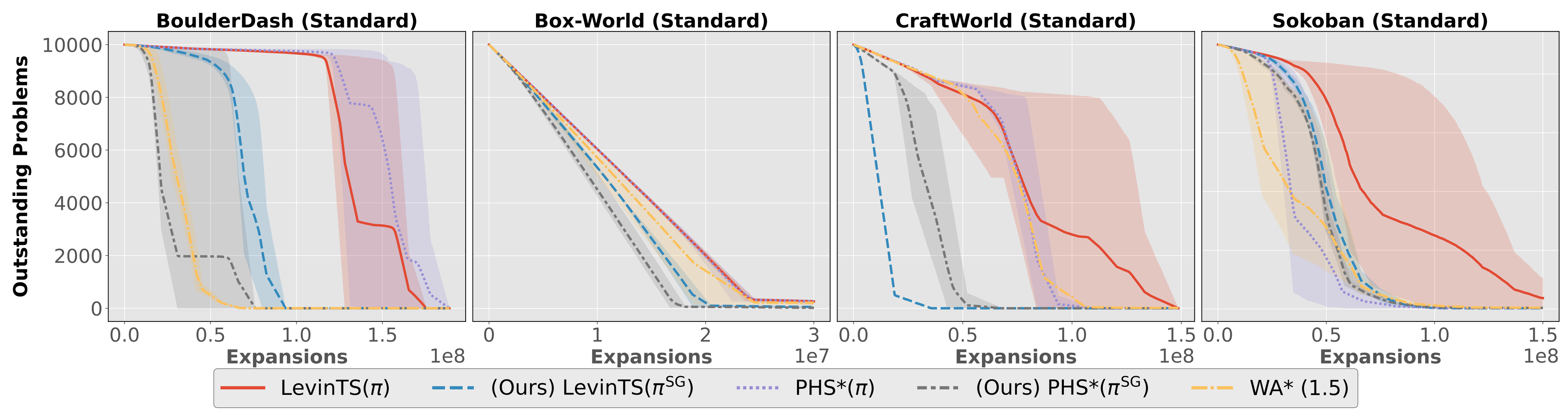


(iv) Update high-level and low-level policy

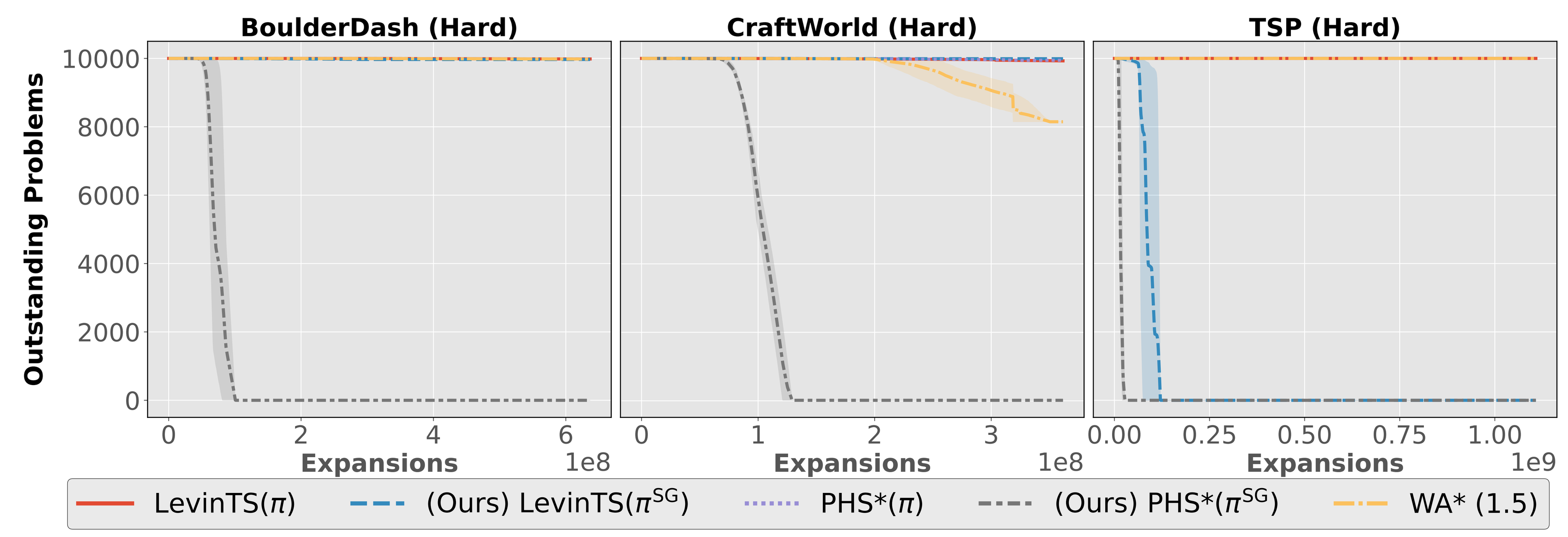


Experiments - Training Search Loss Curves

Standard Problems:



Hard Problems:



Experiment - Test Results

Algorithm	Solved	Expansions	Length
Box-World (Standard)			
WA* (1.5)	100	2,398.68	66.19
LevinTS(π)	100	605.05	67.91
PHS*(π)	100	767.06	68.07
HIPS-ε	100	467.79(†)	67.39
LevinTS(π^{SG})	100	411.38	66.73
PHS*(π^{SG})	100	378.58	66.75
CraftWorld (Standard)			
WA* (1.5)	100	2,318.22	90.01
LevinTS(π)	100	262.76	93.93
PHS*(π)	100	172.04	93.49
HIPS-ε	19	116.11(†)	92.47
LevinTS(π^{SG})	100	208.28	93.93
PHS*(π^{SG})	100	103.23	94.54
BoulderDash (Standard)			
WA* (1.5)	100	1,193.60	51.44
LevinTS(π)	100	61.33	52.90
PHS*(π)	100	53.65	52.74
HIPS-ε	0	—	—
LevinTS(π^{SG})	100	65.48	53.30
PHS*(π^{SG})	100	53.34	52.68
Sokoban (Standard)			
WA* (1.5)	100	1,091.45	32.81
LevinTS(π)	100	1,177.26	41.04
PHS*(π)	100	1,523.38	39.40
HIPS-ε	100	80.55(†)	45.24
LevinTS(π^{SG})	100	496.87	41.85
PHS*(π^{SG})	100	808.42	39.61

Algorithm	Solved	Expansions	Length
CraftWorld (Hard)			
WA* (1.5)	100	313,572.52	117.03
LevinTS(π)	0	—	—
PHS*(π)	0	—	—
LevinTS(π^{SG})	100	395,557.94	123.29
PHS*(π^{SG})	100	3,071.83	120.77
BoulderDash (Hard)			
WA* (1.5)	16	271,636.94	58.19
LevinTS(π)	0	—	—
PHS*(π)	0	—	—
LevinTS(π^{SG})	22	315,807.91	69.5
PHS*(π^{SG})	100	172.32	84.5
TSP (Hard)			
WA* (1.5)	1	502,624.0	45.0
LevinTS(π)	0	—	—
PHS*(π)	0	—	—
LevinTS(π^{SG})	100	570.09	40.89
PHS*(π^{SG})	100	41.93	41.73

Results on the test set for each of the standard and hard domain environments. Expansions and solution length are averaged over solved problems. (†) Expansions reported for HIPS-ε are only for the latent-level space, and thus are not directly comparable to the other algorithms.